

```

1  /*
2  Display.cpp - A simple track GPS to SD card logger. Display module.
3  TinyTrackGPS v0.10
4
5  Copyright © 2019-2021 Francisco Rafael Reyes Carmona.
6  All rights reserved.
7
8  rafael.reyes.carmona@gmail.com
9
10 This file is part of TinyTrackGPS.
11
12 TinyTrackGPS is free software: you can redistribute it and/or modify
13 it under the terms of the GNU General Public License as published by
14 the Free Software Foundation, either version 3 of the License, or
15 (at your option) any later version.
16
17 TinyTrackGPS is distributed in the hope that it will be useful,
18 but WITHOUT ANY WARRANTY; without even the implied warranty of
19 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
20 GNU General Public License for more details.
21
22 You should have received a copy of the GNU General Public License
23 along with TinyTrackGPS. If not, see <https://www.gnu.org/licenses/>.
24 */
25
26 #include "Display.h"
27
28 Display::Display(Display_Type t):_screen(t){
29     //if (_screen == SDD1306_128X64){
30         _width = 16;
31         _height = (_screen > 0) ? 2 : 8;
32         //_offset = 0;
33     } else if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C){
34         // _width = 16;
35         // _height = 2;
36         //_offset = 0;
37     }
38 }
39
40 void Display::start(){
41     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C){
42         #if defined(DISPLAY_TYPE_LCD_16X2)
43             lcd = new LiquidCrystal(LCD_RS, LCD_ENABLE, LCD_D0, LCD_D1, LCD_D2, LCD_D3);
44             lcd->begin(_width, _height);
45             #elif defined(DISPLAY_TYPE_LCD_16X2_I2C)
46                 lcd = new LiquidCrystal_I2C(I2C,_width,_height);
47                 lcd->init();
48                 lcd->backlight();
49             #endif
50
51             #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
52                 // DEFINICION DE CARACTERES PERSONALIZADOS
53                 static byte alt[8] = { 0x04, 0x0E, 0x1F, 0x04, 0x04, 0x04, 0x04, 0x04 };
54                 static byte ant[8] = { 0x0E, 0x11, 0x15, 0x11, 0x04, 0x04, 0x0E, 0x00 };
55                 static byte sd[8] = { 0x0E, 0x11, 0x1F, 0x00, 0x00, 0x17, 0x15, 0x1D };
56                 static byte hourglass_0[8] = { 0x1F, 0x0E, 0x0E, 0x04, 0x04, 0x0A, 0x0A,
57                     0x1F };
58                 static byte hourglass_1[8] = { 0x1F, 0x0A, 0x0E, 0x04, 0x04, 0x0A, 0x0A,
59                     0x1F };

```

```

58     static byte hourglass_2[8] = { 0x1F, 0x0A, 0x0E, 0x04, 0x04, 0x0A, 0x0E,
0x1F };
59     static byte hourglass_3[8] = { 0x1F, 0x0A, 0x0A, 0x04, 0x04, 0x0A, 0x0E,
0x1F };
60     static byte hourglass_4[8] = { 0x1F, 0x0A, 0x0A, 0x04, 0x04, 0x0E, 0x0E,
0x1F };
61     lcd->createChar(0, hourglass_0);
62     lcd->createChar(1, hourglass_1);
63     lcd->createChar(2, hourglass_2);
64     lcd->createChar(3, hourglass_3);
65     lcd->createChar(4, hourglass_4);
66     lcd->createChar(5, alt);
67     lcd->createChar(6, ant);
68     lcd->createChar(7, sd);
69     #endif
70     //}
71
72     //if (_screen == SDD1306_128X64) {
73     #if defined(DISPLAY_TYPE_SDD1306_128X64)
74     u8x8_SSD1306 = new U8X8_SSD1306_128X64_NONAME_HW_I2C(U8X8_PIN_NONE, SCL,
SDA);
75     u8x8_SSD1306->begin();
76     u8x8_SSD1306->setFont(u8x8_font_7x14B_1x2_r);
77     #endif
78     //}
79     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
80     display = new DisplaySSD1306_128x64_I2C(-1);
81     display->begin();
82     //display->setFixedFont(ssd1306xled_font8x16);
83     //display->setFixedFont(ssd1306xled_font6x8);
84     display->setFixedFont(TinyTrackGPS_font8x16);
85     this->clr();
86     #endif
87 }
88
89 void Display::clr(){
90     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
91     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
92     lcd->clear();
93     #endif
94     //}
95     //else if (_screen == SDD1306_128X64) {
96     #if defined(DISPLAY_TYPE_SDD1306_128X64)
97     u8x8_SSD1306->clear();
98     #endif
99     //}
100     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
101     display->clear();
102     #endif
103 }
104
105 void Display::print(int vertical, int horizontal, const char text[]){
106     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
107     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
108     lcd->setCursor(vertical, horizontal);
109     this->print(text);
110     #endif
111     //}
112     //else if (_screen == SDD1306_128X64) {
113     #if defined(DISPLAY_TYPE_SDD1306_128X64)

```

```

114         //u8x8_SSD1306->drawString(vertical, (horizontal*2),text);
115         u8x8_SSD1306->setCursor(vertical, (horizontal*2));
116         this->print(text);
117         #endif
118     //}
119     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
120         display->setTextCursor((vertical*8),(horizontal*16));
121         this->print(text);
122         //display->printFixed((vertical*8),(horizontal*16),text);
123     #endif
124 }
125
126 void Display::print(int line, const char text[]){
127     byte pos = _width -(strlen(text));
128     pos = (pos >> 1);
129     this->print((int)pos, line, text);
130 }
131
132 void Display::print(const char text[]){
133     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
134     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
135         lcd->print(text);
136     #endif
137     //}
138     //else if (_screen == SDD1306_128X64) {
139     #if defined(DISPLAY_TYPE_SDD1306_128X64)
140         u8x8_SSD1306->print(text);
141         u8x8_SSD1306->flush();
142     #endif
143     //}
144     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
145         display->write(text);
146     #endif
147 }
148
149 void Display::print(const char text1[], const char text2[]){
150     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
151         this->print((_screen > 0)?0:1, text1);
152         this->print((_screen > 0)?1:2, text2);
153     //}
154     //else if (_screen == SDD1306_128X64) {
155     //    this->print(1, text1);
156     //    this->print(2, text2);
157     //}
158 }
159
160 void Display::print(const char text1[], const char text2[], const char text3[]){
161     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
162     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
163         this->print(text1, text2);
164         delay(750);
165         //for (unsigned long start = millis(); millis() - start < 750;) {}
166         //unsigned long start = millis();
167         //do {} while (millis() - start < 750);
168
169         this->clr();
170         this->print(0,text3);
171     //}
172     #endif

```

```

173     #if defined(DISPLAY_TYPE_SDD1306_128X64) ||
defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
174     //else if (_screen == SDD1306_128X64) {
175         this->print(0, text1);
176         this->print(1, text2);
177         this->print(2, text3);
178     //}
179     #endif
180 }
181
182 void Display::print(const char text1[], const char text2[], const char text3[],
const char text4[]){
183     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
184     this->print(text1,text2);
185     delay(750);
186     this->print(text3,text4);
187     #endif
188     #if defined(DISPLAY_TYPE_SDD1306_128X64) ||
defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
189     this->print(0, text1);
190     this->print(1, text2);
191     this->print(2, text3);
192     this->print(3, text4);
193     #endif
194 }
195
196 void Display::wait_anin(unsigned int t){
197     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
198     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
199     lcd->setCursor(15,1);
200     lcd->write((byte)t%5);
201     #endif
202     //}
203     //else if (_screen == SDD1306_128X64) {
204     #if defined(DISPLAY_TYPE_SDD1306_128X64)
205
206     const char p[4] = {(char)47,(char)45,(char)92,(char)124};
207     u8x8_SSD1306->setCursor((_width-1),6);
208     u8x8_SSD1306->print(p[t%4]);
209
210     /*
211     static uint8_t hourglass_UP[5][8] = {
0x01,0x1f,0x7f,0xff,0xff,0x7f,0x1f,0x01,
212                                     0x01,0x1f,0x7d,0xf9,0xf9,0x7d,0x1f,0x01,
213                                     0x01,0x1f,0x79,0xf1,0xf1,0x79,0x1f,0x01,
214                                     0x01,0x1f,0x71,0xe1,0xe1,0x71,0x1f,0x01,
215                                     0x01,0x1f,0x61,0x81,0x81,0x61,0x1f,0x01
216                                     };
217
218     static uint8_t hourglass_DOWN[5][8] =
{0x80,0xf8,0x86,0x81,0x81,0x86,0xf8,0x80,
219                                     0x80,0xf8,0xc6,0xe1,0xe1,0xc6,0xf8,0x80,
220                                     0x80,0xf8,0xe6,0xf1,0xf1,0xe6,0xf8,0x80,
221                                     0x80,0xf8,0xfe,0xf9,0xf9,0xfe,0xf8,0x80,
222                                     0x80,0xf8,0xfe,0xff,0xff,0xfe,0xf8,0x80
223                                     };
224     u8x8_SSD1306->drawTile((_width-1), 6, 1, hourglass_UP[t%5]);
225     u8x8_SSD1306->drawTile((_width-1), 7, 1, hourglass_DOWN[t%5]);
226     */
227     #endif

```

```

228 //}
229 #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
230 //const char p[5] = {(char)58,(char)59,(char)60,(char)61,(char)62};
231 display->setTextCursor(120,48);
232 display->printChar((char)(t%5)+58);
233
234 //display->drawWindow(0,0,0,0,"GPS Signal",true);
235 //display->drawProgressBar( t%100 );
236
237 //display->setTextCursor((_width-1)*8,48);
238 //display->printChar('-');
239 #endif
240 }
241
242 void Display::print_PChar(byte c) {
243 //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
244 #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
245 lcd->write(c);
246 #endif
247 //}
248 //else if (_screen == SDD1306_128X64) {
249 #if defined(DISPLAY_TYPE_SDD1306_128X64)
250
251 static uint8_t PChar_UP[3][8] = { 0x30,0x38,0x3c,0xff,0xff,0x3c,0x38,0x30,
252                                     0x3c,0x02,0x01,0xd9,0xd9,0x01,0x02,0x3c,
253                                     0x78,0x7c,0x6e,0x66,0x66,0x6e,0x7c,0x78
254                                     };
255 static uint8_t PChar_DOWN[3][8] = { 0x00,0x00,0x00,0xff,0xff,0x00,0x00,0x00,
256                                     0x00,0xc0,0xe0,0xff,0xff,0xe0,0xc0,0x00,
257                                     0x7c,0xfc,0xc0,0xf8,0x7c,0x0c,0xfc,0xf8
258                                     };
259
260 //char tile;
261 if (c == 5) {
262 //tile = (char)0x18;
263 u8x8_SSD1306->drawTile(9, 2, 1, PChar_UP[0]);
264 u8x8_SSD1306->drawTile(9, 3, 1, PChar_DOWN[0]);
265 //u8x8_SSD1306->setCursor(9, 2);
266 }
267 else if (c == 6) {
268 //tile = (char)0x7f;
269 u8x8_SSD1306->drawTile(11, 0, 1, PChar_UP[1]);
270 u8x8_SSD1306->drawTile(11, 1, 1, PChar_DOWN[1]);
271 //u8x8_SSD1306->setCursor(11, 0);
272 }
273 else if (c == 7) {
274 //tile = (char)0xda;
275 u8x8_SSD1306->drawTile(15, 0, 1, PChar_UP[2]);
276 u8x8_SSD1306->drawTile(15, 1, 1, PChar_DOWN[2]);
277 //u8x8_SSD1306->setCursor(15, 0);
278 }
279 //u8x8_SSD1306->print(tile);
280 #endif
281 //}
282 #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
283 //const char p[3] = {(char)94,(char)95,(char)96};
284 /*
285 switch (c) {
286 case '5':
287 display->setTextCursor(72,32);
288 break;

```

```

288     case '6':
289         display->setTextCursor(88,0);
290         break;
291     case '7':
292         display->setTextCursor(120,0);
293         break;
294     }
295     */
296     /*
297     if( c == 5) {
298         display->setTextCursor(72,32);
299         //display->printChar(p[0]);
300     }
301     else if( c == 6) {
302         display->setTextCursor(88,0);
303         //display->printChar(p[1]);
304     }
305     else if( c == 7) {
306         display->setTextCursor(120,0);
307         //display->printChar(p[2]);
308     }
309     */
310     #ifndef __LGT8F__
311     display->printChar((char)(c+86));
312     #endif
313 #endif
314 }
315
316 void Display::DrawLogo() {
317     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
318     display->drawBitmap1(0,16,128,32,Logo);
319     #endif
320 }
321
322 #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
323 const PROGMEM uint8_t TinyTrackGPS_font8x16[] = {
324     0x00, // 0x00 means fixed font type - the only supported by the library
325     0x08, // 0x08 = 8 - font width in pixels
326     0x10, // 0x10 = 16 - font height in pixels
327     0x2d,
328
329     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01,
330     0x01, 0x01, 0x01, // - 45
331     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00,
332     0x00, 0x00, 0x00, // . 46
333     0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x00, 0x04, 0x04, 0x04, 0x04, 0x04,
334     0x04, 0x04, 0x00, // '/'->'=' 47
335     0x00, 0xE0, 0x10, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x00, 0x0F, 0x10, 0x20, 0x20,
336     0x10, 0x0F, 0x00, // 0 48
337     0x00, 0x10, 0x10, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x20, 0x3F, 0x20,
338     0x20, 0x00, 0x00, // 1 49
339     0x00, 0x70, 0x08, 0x08, 0x08, 0x88, 0x70, 0x00, 0x00, 0x30, 0x28, 0x24, 0x22,
340     0x21, 0x30, 0x00, // 2 50
341     0x00, 0x30, 0x08, 0x88, 0x88, 0x48, 0x30, 0x00, 0x00, 0x18, 0x20, 0x20, 0x20,
342     0x11, 0x0E, 0x00, // 3 51
343     0x00, 0x00, 0xC0, 0x20, 0x10, 0xF8, 0x00, 0x00, 0x00, 0x07, 0x04, 0x24, 0x24,
344     0x3F, 0x24, 0x00, // 4 52
345     0x00, 0xF8, 0x08, 0x88, 0x88, 0x08, 0x08, 0x00, 0x00, 0x19, 0x21, 0x20, 0x20,
346     0x11, 0x0E, 0x00, // 5 53

```

```
338 0x00, 0xE0, 0x10, 0x88, 0x88, 0x18, 0x00, 0x00, 0x00, 0x0F, 0x11, 0x20, 0x20,
0x11, 0x0E, 0x00, // 6 54
339 0x00, 0x38, 0x08, 0x08, 0xC8, 0x38, 0x08, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x00,
0x00, 0x00, 0x00, // 7 55
340 0x00, 0x70, 0x88, 0x08, 0x08, 0x88, 0x70, 0x00, 0x00, 0x1C, 0x22, 0x21, 0x21,
0x22, 0x1C, 0x00, // 8 56
341 0x00, 0xE0, 0x10, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x00, 0x00, 0x31, 0x22, 0x22,
0x11, 0x0F, 0x00, // 9 57
342
343 0x01,0x1f,0x7f,0xff,0xff,0x7f,0x1f,0x01,0x80,0xf8,0x86,0x81,0x81,0x86,0xf8,0x80,
// Code for char num 58 :
344 0x01,0x1f,0x7d,0xf9,0xf9,0x7d,0x1f,0x01,0x80,0xf8,0xc6,0xe1,0xe1,0xc6,0xf8,0x80,
// Code for char num 59 ;
345 0x01,0x1f,0x79,0xf1,0xf1,0x79,0x1f,0x01,0x80,0xf8,0xe6,0xf1,0xf1,0xe6,0xf8,0x80,
// Code for char num 60 <
346 0x01,0x1f,0x71,0xe1,0xe1,0x71,0x1f,0x01,0x80,0xf8,0xfe,0xf9,0xf9,0xfe,0xf8,0x80,
// Code for char num 61 =
347 0x01,0x1f,0x61,0x81,0x81,0x61,0x1f,0x01,0x80,0xf8,0xfe,0xff,0xff,0xfe,0xf8,0x80,
// Code for char num 62 >
348
349 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
// Code for char num 63 ? -> ' '
350 0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x00,0x20,0x3F,0x20,0x00,0x3F,0x20,0x00,0x3F,
// Code for char num 64 @ -> 'm'
351
352 0x00, 0x00, 0xC0, 0x38, 0xE0, 0x00, 0x00, 0x00, 0x20, 0x3C, 0x23, 0x02, 0x02,
0x27, 0x38, 0x20, // A 33
353 0x08, 0xF8, 0x88, 0x88, 0x88, 0x70, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x20,
0x11, 0x0E, 0x00, // B 34
354 0xC0, 0x30, 0x08, 0x08, 0x08, 0x08, 0x38, 0x00, 0x07, 0x18, 0x20, 0x20, 0x20,
0x10, 0x08, 0x00, // C 35
355 0x08, 0xF8, 0x08, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x20,
0x10, 0x0F, 0x00, // D 36
356 0x08, 0xF8, 0x88, 0x88, 0xE8, 0x08, 0x10, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x23,
0x20, 0x18, 0x00, // E 37
357 0x08, 0xF8, 0x88, 0x88, 0xE8, 0x08, 0x10, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x03,
0x00, 0x00, 0x00, // F 38
358 0xC0, 0x30, 0x08, 0x08, 0x08, 0x38, 0x00, 0x00, 0x07, 0x18, 0x20, 0x20, 0x22,
0x1E, 0x02, 0x00, // G 39
359 0x08, 0xF8, 0x08, 0x00, 0x00, 0x08, 0xF8, 0x08, 0x20, 0x3F, 0x21, 0x01, 0x01,
0x21, 0x3F, 0x20, // H 40
360 0x00, 0x08, 0x08, 0xF8, 0x08, 0x08, 0x00, 0x00, 0x00, 0x20, 0x20, 0x3F, 0x20,
0x20, 0x00, 0x00, // I 41
361 0x00, 0x00, 0x08, 0x08, 0xF8, 0x08, 0x08, 0x00, 0xC0, 0x80, 0x80, 0x80, 0x7F,
0x00, 0x00, 0x00, // J 42
362 0x08, 0xF8, 0x88, 0xC0, 0x28, 0x18, 0x08, 0x00, 0x20, 0x3F, 0x20, 0x01, 0x26,
0x38, 0x20, 0x00, // K 43
363 0x08, 0xF8, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x20,
0x20, 0x30, 0x00, // L 44
364 0x08, 0xF8, 0xF8, 0x00, 0xF8, 0xF8, 0x08, 0x00, 0x20, 0x3F, 0x00, 0x3F, 0x00,
0x3F, 0x20, 0x00, // M 45
365 0x08, 0xF8, 0x30, 0xC0, 0x00, 0x08, 0xF8, 0x08, 0x20, 0x3F, 0x20, 0x00, 0x07,
0x18, 0x3F, 0x00, // N 46
366 0xE0, 0x10, 0x08, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x0F, 0x10, 0x20, 0x20, 0x20,
0x10, 0x0F, 0x00, // O 47
367 0x08, 0xF8, 0x08, 0x08, 0x08, 0x08, 0xF0, 0x00, 0x20, 0x3F, 0x21, 0x01, 0x01,
0x01, 0x00, 0x00, // P 48
368 0xE0, 0x10, 0x08, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x0F, 0x18, 0x24, 0x24, 0x38,
0x50, 0x4F, 0x00, // Q 49
```

```

369     0x08, 0xF8, 0x88, 0x88, 0x88, 0x88, 0x70, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x03,
    0x0C, 0x30, 0x20, // R 50
370     0x00, 0x70, 0x88, 0x08, 0x08, 0x08, 0x38, 0x00, 0x00, 0x38, 0x20, 0x21, 0x21,
    0x22, 0x1C, 0x00, // S 51
371     0x18, 0x08, 0x08, 0xF8, 0x08, 0x08, 0x18, 0x00, 0x00, 0x00, 0x20, 0x3F, 0x20,
    0x00, 0x00, 0x00, // T 52
372     0x08, 0xF8, 0x08, 0x00, 0x00, 0x08, 0xF8, 0x08, 0x00, 0x1F, 0x20, 0x20, 0x20,
    0x20, 0x1F, 0x00, // U 53
373     0x08, 0x78, 0x88, 0x00, 0x00, 0xC8, 0x38, 0x08, 0x00, 0x00, 0x07, 0x38, 0x0E,
    0x01, 0x00, 0x00, // V 54
374     0xF8, 0x08, 0x00, 0xF8, 0x00, 0x08, 0xF8, 0x00, 0x03, 0x3C, 0x07, 0x00, 0x07,
    0x3C, 0x03, 0x00, // W 55
375     0x08, 0x18, 0x68, 0x80, 0x80, 0x68, 0x18, 0x08, 0x20, 0x30, 0x2C, 0x03, 0x03,
    0x2C, 0x30, 0x20, // X 56
376     0x08, 0x38, 0xC8, 0x00, 0xC8, 0x38, 0x08, 0x00, 0x00, 0x00, 0x20, 0x3F, 0x20,
    0x00, 0x00, 0x00, // Y 57
377     0x10, 0x08, 0x08, 0x08, 0xC8, 0x38, 0x08, 0x00, 0x20, 0x38, 0x26, 0x21, 0x20,
    0x20, 0x18, 0x00, // Z 58
378
379
    0x30,0x38,0x3c,0xff,0xff,0x3c,0x38,0x30,0x00,0x00,0x00,0xff,0xff,0x00,0x00,0x00,//
Code for char num 91 '['
380
    0x3c,0x02,0x01,0xd9,0xd9,0x01,0x02,0x3c,0x00,0xc0,0xe0,0xff,0xff,0xe0,0xc0,0x00,//
Code for char num 92 '\'
381
    0x78,0x7c,0x6e,0x66,0x66,0x6e,0x7c,0x78,0x7c,0xfc,0xc0,0xf8,0x7c,0x0c,0xfc,0xf8,//
Code for char num 93 ']'
382 /*
383     0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x19, 0x24, 0x22, 0x22,
    0x22, 0x3F, 0x20, // a 65
384     0x08, 0xF8, 0x00, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x11, 0x20, 0x20,
    0x11, 0x0E, 0x00, // b 66
385     0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x0E, 0x11, 0x20, 0x20,
    0x20, 0x11, 0x00, // c 67
386     0x00, 0x00, 0x00, 0x80, 0x80, 0x88, 0xF8, 0x00, 0x00, 0x0E, 0x11, 0x20, 0x20,
    0x10, 0x3F, 0x20, // d 68
387     0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x1F, 0x22, 0x22, 0x22,
    0x22, 0x13, 0x00, // e 69
388     0x00, 0x80, 0x80, 0xF0, 0x88, 0x88, 0x88, 0x18, 0x00, 0x20, 0x20, 0x3F, 0x20,
    0x20, 0x00, 0x00, // f 70
389     0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x6B, 0x94, 0x94, 0x94,
    0x93, 0x60, 0x00, // g 71
390     0x08, 0xF8, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x20, 0x3F, 0x21, 0x00, 0x00,
    0x20, 0x3F, 0x20, // h 72
391     0x00, 0x80, 0x98, 0x98, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x20, 0x3F, 0x20,
    0x20, 0x00, 0x00, // i 73
392     0x00, 0x00, 0x00, 0x80, 0x98, 0x98, 0x00, 0x00, 0x00, 0xC0, 0x80, 0x80, 0x80,
    0x7F, 0x00, 0x00, // j 74
393     0x08, 0xF8, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0x20, 0x3F, 0x24, 0x02, 0x2D,
    0x30, 0x20, 0x00, // k 75
394     0x00, 0x08, 0x08, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x20, 0x3F, 0x20,
    0x20, 0x00, 0x00, // l 76
395     0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x3F,
    0x20, 0x00, 0x3F, // m 77
396     0x80, 0x80, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x20, 0x3F, 0x21, 0x00, 0x00,
    0x20, 0x3F, 0x20, // n 78
397     0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x1F, 0x20, 0x20, 0x20,
    0x20, 0x1F, 0x00, // o 79

```



```

398     0x80, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x00, 0x80, 0xFF, 0xA1, 0x20, 0x20,
    0x11, 0x0E, 0x00, // p 80
399     0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x0E, 0x11, 0x20, 0x20,
    0xA0, 0xFF, 0x80, // q 81
400     0x80, 0x80, 0x80, 0x00, 0x80, 0x80, 0x80, 0x00, 0x20, 0x20, 0x3F, 0x21, 0x20,
    0x00, 0x01, 0x00, // r 82
401     0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x33, 0x24, 0x24, 0x24,
    0x24, 0x19, 0x00, // s 83
402     0x00, 0x80, 0x80, 0xE0, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x20,
    0x20, 0x00, 0x00, // t 84
403     0x80, 0x80, 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00, 0x1F, 0x20, 0x20, 0x20,
    0x10, 0x3F, 0x20, // u 85
404     0x80, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0x01, 0x0E, 0x30, 0x08,
    0x06, 0x01, 0x00, // v 86
405     0x80, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x80, 0x0F, 0x30, 0x0C, 0x03, 0x0C,
    0x30, 0x0F, 0x00, // w 87
406     0x00, 0x80, 0x80, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x20, 0x31, 0x2E, 0x0E,
    0x31, 0x20, 0x00, // x 88
407     0x80, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x81, 0x8E, 0x70, 0x18,
    0x06, 0x01, 0x00, // y 89
408     0x00, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x21, 0x30, 0x2C, 0x22,
    0x21, 0x30, 0x00, // z 90
409 */
410 };
411
412 //-----
413 // File generated by LCD Assistant
414 // http://en.radzio.dxp.pl/bitmap_converter/
415 //-----
416
417 const PROGMEM uint8_t Logo[] = {
418 0x00, 0x00, 0x80, 0xC0, 0x60, 0x10, 0x98, 0x4C, 0x64, 0x26, 0x12, 0x12, 0x0B, 0x09,
    0x09, 0x49,
419 0x49, 0x09, 0x09, 0x0B, 0x12, 0x12, 0x26, 0x64, 0x48, 0x98, 0x10, 0x60, 0xC0, 0x00,
    0x00, 0x00,
420 0x00, 0x00, 0x00, 0x18, 0x08, 0xF8, 0x08, 0x08, 0x00, 0x00, 0x00, 0x00, 0xE8, 0x00,
    0x00, 0x00,
421 0x00, 0x00, 0xE0, 0x20, 0x20, 0xE0, 0x00, 0x00, 0x00, 0x20, 0xE0, 0x00, 0xC0, 0x20,
    0x00, 0x00,
422 0x00, 0x00, 0x18, 0x08, 0xF8, 0x08, 0x08, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x20, 0x20,
    0x00, 0x00,
423 0x00, 0x00, 0x40, 0x60, 0xA0, 0xE0, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x20, 0x20, 0x60,
    0x00, 0x00,
424 0x00, 0xF8, 0x80, 0xC0, 0x20, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x18, 0x08, 0x08, 0x98,
    0x00, 0x00,
425 0x00, 0x00, 0xF8, 0x48, 0x48, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x68, 0x48,
    0x98, 0x00,
426 0xF0, 0x1E, 0x03, 0xF0, 0x0C, 0x03, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0,
    0x60, 0xB0,
427 0x10, 0x18, 0x08, 0xC4, 0x64, 0x1E, 0x07, 0x03, 0x00, 0x81, 0x03, 0x0C, 0xF0, 0x03,
    0x1E, 0xF0,
428 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x02,
    0x00, 0x00,
429 0x00, 0x00, 0x03, 0x02, 0x00, 0x03, 0x02, 0x00, 0x00, 0x08, 0x08, 0x07, 0x00, 0x00,
    0x00, 0x00,
430 0x00, 0x00, 0x00, 0x00, 0x03, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x02, 0x00,
    0x00, 0x00,
431 0x00, 0x00, 0x03, 0x02, 0x02, 0x03, 0x02, 0x00, 0x00, 0x00, 0x01, 0x02, 0x02, 0x02,
    0x00, 0x00,

```

[illegible]